

\mathcal{P} is not equal to \mathcal{NP} .

Sten-Åke Tärnlund^{*†‡}

July 13, 2009

Abstract

$SAT \notin \mathcal{P}$ is true, and provable in the simply consistent extension \mathbf{B}' of the first order theory \mathbf{B} of computing, with the single finite axiom B characterizing a universal Turing machine. Therefore $\mathcal{P} \neq \mathcal{NP}$ is true, and provable in the simply consistent extension \mathbf{B}'' of theory \mathbf{B} .

1 Introduction

$SAT \notin \mathcal{P}$ is true, theorem 1, and provable, corollary 9, in the simply consistent extension \mathbf{B}' of the first order theory \mathbf{B} of computing, with a single finite axiom B characterizing a universal Turing machine.¹

Therefore, $\mathcal{P} \neq \mathcal{NP}$ is true, theorem 2, and provable, corollary 10, in the simply consistent extension \mathbf{B}'' of theory \mathbf{B} , by the Cook-Levin theorem.^{2 3} SAT is the set of satisfiable propositional formulas. \mathcal{P} is the set of classes of problems with solutions in polynomial computing time, for a deterministic Turing machine, in contrast, \mathcal{NP} is the corresponding set for a Turing machine.⁴

The proof of theorem 1 is a proof by contradiction in the extension \mathbf{B}' of theory \mathbf{B} , using the axiomatic method.^{5 6} \mathbf{B}' is simply consistent⁷ in the subset \mathcal{U} of deterministic Turing machines that decide whether a propositional formula is satisfiable or not, corollary 2.⁸ This justifies the indirect proof method.

^{*}gmail: stenake.

[†] 2nd printing. © Sten-Åke Tärnlund, 2009.

[‡]In the 2nd printing the proof, in the 1st printing, of theorem 1 is divided into three parts a new lemma 4, a new corollary 8, and the remaining part of the original proof. There is no appendix in the 2nd printing, but it is available at www.arXiv.0810.5056v1. The 2nd printing contains some simplifications, more explanations, but no error has been corrected. The main results were first presented 6 August 2008 in a seminar at the department of Information Science of Uppsala University Sweden, www.uu.se.

¹Following Tärnlund 2008 [23], a simplification of Tärnlund 1977 [22], cf. Turing 1936 [25] and Kleene 1967 [12].

² $SAT \in \mathcal{P} \equiv \mathcal{P} = \mathcal{NP}$, Cook 1971 [3] and Levin 1973 [14], cf. Sipser 2005 [20].

³ \mathcal{P} vs \mathcal{NP} cf. Smale 1998 [21], Cook 2003 [4], and Cook www.claymath.org/millennium.

⁴cf. Karp 1972 [11], and Sipser 2005 [20].

⁵Hilbert and Bernays 1934 [10], cf. Kleene 1967 [12].

⁶The axiomatic method is useful also in computing, e.g. proving that a program is correct, Clark and Tärnlund 1977 [1], and computability, Tärnlund 2008 [23].

⁷Simple consistency, cf. Kleene 1967 [12].

⁸For the set of all Turing machines, however, theory \mathbf{B} is simply consistent if and only if simple consistency of \mathbf{B} is not provable in \mathbf{B} , Tärnlund 2008 [23].

The idea of the proof of theorem 1, i.e. $SAT \notin \mathcal{P}$, is a relationship between computing time,⁹ and proof complexity¹⁰ in theory **B**.

By axiom B , for a Turing machine in \mathcal{U} , which decides whether or not $\neg F$ is satisfiable for a sufficiently large tautology F ³⁹ (on disjunctive normal form), there is a deduction in theory **B** that is a propositional formula:

$$(Q_0 \supset F) \wedge (Q_1 \supset Q_0) \wedge \cdots \wedge (Q_n \supset Q_{n-1}) \wedge Q_n, \quad (1)$$

where Q_0, \dots, Q_n are atoms, and n expresses the number of moves of the tape-head of the Turing machine (computing time).

For several deduction systems including Robinson's resolution,^{11 12} (1) yields a formal deduction of F , i.e.

$$Q_n, \neg Q_n \vee Q_{n-1}, Q_{n-1}, \dots, Q_1, \neg Q_1 \vee Q_0, Q_0, \neg Q_0 \vee F, F. \quad (2)$$

Thus, (2) is a proof of F in resolution system R , i.e. $\vdash_R F$. Further, the size of the resolution deduction of F is the number of symbols in (2),¹⁰ cf. lemma 4.

Now, assume that

$$SAT \in \mathcal{P}. \quad (3)$$

Then, there is a Turing machine in \mathcal{U} that decides whether $\neg F$ is satisfiable or not in polynomial time in the size of F , i.e. $p(F)$. Thus for (1) - (2),

$$n \leq p(F) \text{ for some } i \in \mathcal{U} \text{ any sufficiently large tautology } F. \quad (4)$$

In addition (2) gives,

$$\vdash_R F \text{ and the size of (2)} \leq p(F). \quad (5)$$

Now there is a contradiction by (5) and the existence of sufficiently large tautologies not having a resolution proof in polynomial size, e.g. valid pigeonhole formulas,¹³ Haken's theorem,¹⁴ and Razborov's corollary.¹⁵

Thus, assumption (3) is false, this explanation has arrived at theorem 1:

$$SAT \notin \mathcal{P}. \quad (6)$$

Therefore theorem 2: $\mathcal{P} \neq \mathcal{NP}$, follows by the Cook-Levin theorem.²

It remains to present the crucial axiom B , and the first order theory **B** of computing in section 2. Notions of complexity are introduced in sections 3 and 4. Essentially, axiom B gives lemma 3, which yields lemma 4. Then, theorem 1 is reached readily in the extension **B'** of theory **B**.

⁹ The number moves of the tapehead of a Turing machine, with an output, definition 3, cf. Hartmanis and Stearns 1965 [9], and Sipser 2005 [20].

¹⁰ The size of a formal deduction (proof) in propositional logic, cf. definition 18.

¹¹ Robinson 1965 [19].

¹² The resolution system R is used. This deduction system is defined in (47) - (50).

¹³ PHF_n^m for a propositional pigeonhole formula with n holes and m pigeons, valid or not, where $m, n \in \mathbb{N}$. The pigeonhole principle is: there is no injective function with a smaller co-domain than domain for finite sets. This idea can be expressed as a valid pigeonhole formula in disjunctive normal form, i.e. PHP_n^m for $\models PHF_n^m$ some $m, n \in \mathbb{N}$. Some special cases of PHP_n^m : $m = n + 1$ is classic (it also has the name PF_n , cf. Cook and Reckhow 1979 [5]), $m \geq 2n$ is weak, $m = n^2$ is very weak, and $m \rightarrow \infty$ is still weaker.

¹⁴ Haken's theorem 1985 [8] for the classic pigeonhole tautology PF_n : every resolution proof of PF_n contains at least c^n different clauses for $c > 1$ some $c \in \mathbb{R}$ any sufficiently large $n \in \mathbb{N}$.

¹⁵ A pigeonhole tautology PHP_n^m , as a function, has a lower bound $\exp(\Omega(n^{1/3}))$ of the size of the proof in resolution, for an arbitrary $m > n$. This is Razborov's corollary 2003 [18].

2 A theory of computing

The first order theory **B** of computing with a single finite axiom B , which characterizes a universal Turing machine, is presented in this section.¹⁶ Complexity concepts are introduced in sections 3 and 4.

Syntactically, there are two predicate symbols of theory **B** written $T(i, a, u)$, and $U(x, s, z, q, j, i, u)$. In addition, there is one function symbol $.$ in infix notation $x.y$ representing a list.

A Turing machine has a finite supply of arbitrary constant symbols, e.g. the alphabetic symbols, the natural numbers, and the symbols of propositional logic. For convenience, there is at least a subset of symbols,

$$\{\emptyset, 0, 1, \sqcup\} \subseteq K, \quad (7)$$

where K is a finite set of constant symbols, and \sqcup a blank symbol.

There are six sets for the Turing machines in theory **B**.

A finite set of states,¹⁷

$$Q \subset N, \quad (8)$$

where 0 is the halt state and 1 is the start state.

A finite set S of symbols,¹⁸

$$S \text{ for } \{u : u \in K \vee (u = r.\emptyset \vee u = \emptyset.r) \wedge r \in K\}. \quad (9)$$

The set D of moves of the tapehead of a Turing machine,

$$D \text{ for } \{0, 1\}, \quad (10)$$

where 0 is a move to the left and 1 a move to the right.

There is a finite arbitrary large two-way tape, with a left and right tape having an element between them at the tapehead.¹⁹ Initially, the two-way tape has an empty left tape, the input on the right tape, and the element between them has the symbol \emptyset . When a computation starts the tapehead reads the symbol \emptyset . The arbitrary long but finite two-way tape is represented as two lists.²⁰

The list on the right tape grows to the right, if the symbol $r.\emptyset \in S$ substitutes \emptyset . The list on the left tape grows to the left, if the symbol $\emptyset.r \in S$ substitutes \emptyset . Therefore the size of the two-way tape grows one element at a time controlled by the Turing machine.

There are two sets of lists for the two-way tapes, L for the right tapes, and L' for the left tapes. They are as follows.

¹⁶The postulates of predicate calculus are employed in theory **B**. Frequently, $G4$ of Kleene 1967 [12], a Gentzen-type system 1934-5 [7] is used. The notation can simply be changed to a Hilbert-type system by Gentzen's theorem Kleene 1967 [12], and to a resolution system by Robinson's theorem 1965 [19]. Similarly, for systems in Quine 1974 § 37 [17].

¹⁷ N is the set of the natural numbers.

¹⁸ Operators: $\supset, \equiv, \vee, \wedge, \neg, \in, =, \leq, \forall, \exists, \vdash, \models, \rightarrow$, are ranked decreasingly to get simpler expressions. Thus, $\vdash B \rightarrow P \wedge F$ shall mean $(\vdash (B \rightarrow P)) \wedge F$. Moreover, the operators are used autonomously Kleene 1967 [12].

¹⁹Turing 1936 [25] has a one-way tape, for a two-way tape cf. Post 1947 [16].

²⁰Historically, Turing 1936 [25] and Kleene 1967 [12] have a potentially infinite tape. In contrast, Davis 1958 [6] and Minsky 1967 [15] grow the arbitrary large finite tape in the computation.

The right tapes are lists of symbols,²¹

$$L(\emptyset) \wedge \forall (s \in S \wedge L(z) \supset L(s.z)). \quad (11)$$

The left tapes are lists of symbols,

$$L'(\emptyset) \wedge \forall (s \in S \wedge L'(z) \supset L'(z.s)). \quad (12)$$

The set L of the right tapes,

$$L \text{ for } \{u : L(u)\}. \quad (13)$$

The set L' of the left tapes,

$$L' \text{ for } \{u : L'(u)\}. \quad (14)$$

The set $M \subset L$ of codes of Turing machines is a subset of the set of lists. The code of a Turing machine is a list of quintuples,²²

$$M(\emptyset) \wedge \forall (p, q \in Q \wedge r, s \in S \wedge d \in D \wedge M(z) \supset M(p.s.q.r.d.z)). \quad (15)$$

The set M of codes of Turing machines,

$$M \text{ for } \{u : M(u)\}. \quad (16)$$

The formulas of **B** are defined as usual in a first order theory.²³

Semantically, the infix function symbol and the predicate symbols denote two functions²⁴ and two relations²⁵. Of course, there is an intended interpretation of the function symbol and the predicates in the intended domains of theory **B**.

$T(i, a, u)$ shall mean Turing machine i with input a computes an output u and halts for $i \in M, a \in L, u \in L'$.^{26 27}

$U(x, s, z, q, j, i, u)$ shall mean Turing machine i computes u and halts, where $x.s.z$ is the two-way tape of i ,¹⁹ s is a symbol (at the tapehead), x is the left tape, z is the right tape, i is in state q , and has an auxiliary code j , for $i, j \in M, s \in S, z \in L, q \in Q, x, u \in L'$.

For instance, if $T(i, A.\emptyset, \emptyset.\emptyset.A.B)$ then $U(\emptyset, \emptyset, A.\emptyset.\emptyset, 1, i, i, \emptyset.\emptyset.A.B)$, i.e. Turing machine i computes the output $\emptyset.\emptyset.A.B$. It starts in state 1, reads the symbol \emptyset (by its tapehead), and the two-way tape $\emptyset, \emptyset, A.\emptyset.\emptyset$ is input. Here, the left tape is the empty list \emptyset , and the right tape is the list $A.\emptyset.\emptyset$, where $A.\emptyset$ is the input list.²⁸ Thus, the output added one element B to the input list.

The axiom of theory **B** has the name B .^{1 29} It is first written down, then an informal explanation of B follows.³⁰

²¹ $\forall F$ for a free variable is universally quantified over the entire formula F .

²² cf. Turing 1936 [25].

²³cf. Kleene 1967 [12].

²⁴There are two functions with similar syntax. (i) $.(r, z) \mapsto r.z$, where $r \in S$ and $z \in L$. (ii) $.(x, r) \mapsto x.r$, where $r \in S$ and $x \in L'$. They are distinguished by their appearance, function (i) on the right tape and (ii) on the left tape.

²⁵ $T \subseteq M \times L \times L'$, and $U \subseteq L \times S \times L \times Q \times M \times M \times L'$.

²⁶For example, the formula: $A \supset B$, is written: $A. \supset .B.\emptyset$, as an input list.

²⁷cf. Kleene p. 243 and footnote 167 Kleene 1967 [12].

²⁸Note that the input list $A.\emptyset$ is represented as $A.\emptyset.\emptyset$ on the right tape of the two-way tape, i.e. a list on a list. Thus, the beginning and end of the input list are marked with the symbol \emptyset . The beginning and end of the two-way tape itself are also marked with \emptyset .

²⁹The universal Turing machine in axiom B is also a logic program, i.e. for a Turing machine $i \in M$ with input $a \in L$ and output $u \in L'$, B computes u , e.g. there is a deduction of u in Prolog, cf. Kowalski 1974 [13], Colmerauer et al 1973 [2], and Warren 1977 [26].

³⁰The appendix of Tärnlund 2008 [24] shows examples of computations from axiom B .

Axiom 1 B for

$$\forall T(i, a, u) \supset U(\emptyset, \emptyset, a.\emptyset, 1, i, i, u). \quad (17)$$

$$\forall U(\emptyset, \emptyset, a.\emptyset, 1, i, i, u) \supset T(i, a, u). \quad (18)$$

$$\forall U(x, s, z, 0, i, i, x). \quad (19)$$

$$\forall U(x, v, r.z, p, i, i, u) \supset U(x.v, s, z, q, q.s.p.r.0, j, i, u). \quad (20)$$

$$\forall U(x.r, v, z, p, i, i, u) \supset U(x, s, v.z, q, q.s.p.r.1, j, i, u). \quad (21)$$

$$\forall U(x, s, z, q, j, i, u) \supset U(x, s, z, q, q'.s'.p.r.d, j, i, u). \quad (22)$$

Here, \emptyset , 0 and 1 are constant symbols, and $.$ a function symbol.²⁴

The intended domains are the sets Q in (8), S in (9), D in (10), L in (13), L' in (14), and M in (16).

In sentences (17) - (18), there is an equivalence between Turing machine i with input a and output u , and a universal Turing machine computing output u with the concrete representations of: the quintuples of i , the two-way tape holding input a and the symbol \emptyset , the state 1 , and the auxiliary code i .

There is a halt condition in sentence (19), thus for state 0 the left tape x is the output of the computation that stops.

In sentence (20), if there is a new configuration (the tapehead of Turing machine i has printed r on the tape, moved to the left, and entered state p) then in the previous configuration i is in state q reading symbol s and has the quintuple $q.s.p.r.0$.

Similarly in sentence (21), if there is a new configuration (the tapehead of Turing machine i has printed r on the tape, moved to the right, and entered state p) then in the previous configuration i is in state q reading symbol s and has the quintuple $q.s.p.r.1$.

In sentence (22), Turing machine i is in state q , reads symbol s , and searches for a quintuple $q.s.p.r.d$.

Now, if $\exists uT(i, a, u)$ is true then $\exists uT(i, a, u)$ is provable from axiom 1, by induction on the number of moves of the tapehead in a proof.^{31 32}

Lemma 1 $\exists uT(i, a, u) \supset \vdash B \rightarrow \exists uT(i, a, u)$ any $i \in M$ $a \in L$.

The notion of a Turing machine computation as a formal proof in theory \mathbf{B} , is justified by axiom 1, and lemma 1.^{33 30}

Definition 1 A Turing machine computation for a formal proof in theory \mathbf{B} .

The set \mathcal{D} of deterministic Turing machines is a proper subset of the set of all Turing machines, and is introduced next.

Definition 2 \mathcal{D} for $\{i : i \text{ has no quintuples } q.s.p.r.d \text{ and } q.s.p'.r'.d' \text{ and } \neg(p = p' \wedge r = r' \wedge d = d') \text{ and } i \in M\}$.

³¹The quantifiers some and any in the proof (meta) theory of \mathbf{B} are applied to the entire sentence in front of them.

³²There is a proof of lemma 1 in Tärnlund 2008 [23].

³³For a formal proof in predicate calculus, cf. Kleene 1967 [12].

3 Computing time

In this section some complexity notions are introduced.

For a Turing machine, with an output, the computing time is the number of moves of its tapehead in the computation.⁹ A Turing machine computation is a formal proof in theory **B**, thus the computing time is the number of moves of the tapehead in a proof in **B**.^{16 33 34 35}

Definition 3 $H(\vdash B \rightarrow \exists uT(i, a, u), n)$ for n is the number of moves of the tapehead of Turing machine i in a proof of the sequent $B \rightarrow \exists uT(i, a, u)$ in $G4$ any $i \in M$ $a \in L$ $n \in N$.

The function $||$ computes the size of an (input) list.³⁶

Definition 4 $|a|$ for the number of symbols of $a \in L$.

A polynomial in the size of (the input) a is written $p(a)$.

Definition 5 $p(a)$ for $c \cdot |a|^q$ some $c \in N$ any $a \in L$.

The notion of a polynomial upper bound of the computing time is introduced.

Definition 6 $\vdash B \rightarrow \exists uT(i, a, u)$ in $p(a)$ for $H(\vdash B \rightarrow \exists uT(i, a, u), n) \wedge n \leq p(a)$ any $i \in M$ $a \in L$ some $n \in N$.

Definition 7 \mathcal{F} for the set of formulas of propositional logic.

The set SAT of satisfiable propositional formulas, and the set $TAUT$ of tautologies are introduced.

Definition 8 SAT for $\{F : F \in \mathcal{F} \wedge \not\models \neg F\}$.

Definition 9 $TAUT$ for $\{F : F \in \mathcal{F} \wedge \models F\}$.

A name s is introduced for a Turing machine that computes whether a propositional formula²⁶ is satisfiable, with output $\emptyset.0$, or not with output $\emptyset.1$.³⁷

Definition 10 $T(s, a, u)$ for $a = F.\emptyset \wedge ((u = \emptyset.0 \wedge \not\models \neg F) \vee (u = \emptyset.1 \wedge \models \neg F))$ $s \in M$ any $a \in L$ $F \in \mathcal{F}$ $u \in L$.

The set of correct deterministic Turing machines that decide whether a propositional formula is satisfiable or not, is specified by Turing machine s , by definitions 2 and 10.

Definition 11 \mathcal{U} for $\{i : \exists u (T(i, a, u) \wedge T(s, a, u)) \wedge i \in \mathcal{D} \wedge \text{any } a \in L\}$.

$T(i, \neg F.\emptyset, \emptyset.1)$ if and only if F is a tautology for any i in \mathcal{U} and F in \mathcal{F} .

Corollary 1 $T(i, \neg F.\emptyset, \emptyset.1) \equiv \models \rightarrow F$ any $i \in \mathcal{U}$ $F \in \mathcal{F}$.

³⁴A move of the tapehead of a Turing machine is specified in (20) - (21) of axiom 1.

³⁵Clearly, there is a relation that counts the number of moves of the tapehead of a Turing machine in a proof in theory **B**, i.e. $H \subseteq E \times N$, where E is the set of proofs in theory **B** and N is the set of natural numbers.

³⁶ $|| : L \rightarrow N$.

³⁷Of course, Turing machine s could compute the output satisfiable for $\emptyset.0$, and unsatisfiable for $\emptyset.1$. For reasons of space, the shorter version is used.

Theory **B** is simply consistent in \mathcal{U} , i.e. there is no contradiction, by axiom 1, lemma 1, and definitions 10 - 11.^{7 8}

Corollary 2 $\vdash B \rightarrow (\exists u T(i, a, u) \wedge \neg \exists u T(i, a, u))$ no $i \in \mathcal{U}$ $a \in L$.

In particular, $\exists u T(i, a, u)$ is true if and only if $\exists u T(i, a, u)$ is provable from axiom B for $i \in \mathcal{U}$ $a \in L$, by axiom 1, corollary 2, lemma 1, and definition 11.

Corollary 3 $\exists u T(i, a, u) \equiv \vdash B \rightarrow \exists u T(i, a, u)$ any $i \in \mathcal{U}$ $a \in L$.

The formula $SAT \in \mathcal{P}$ is introduced in the simply consistent extension **B'** of theory **B**, corollary 2. Writing $SAT \in \mathcal{P}$ for there is some deterministic Turing machine in \mathcal{U} that decides, in polynomial time, whether any propositional formula is satisfiable or not, cf. footnotes 2 - 4.

Definition 12 $SAT \in \mathcal{P}$ for $\vdash B \rightarrow \exists u T(i, F \cdot \emptyset, u)$ in $p(F)$ some $i \in \mathcal{U}$ any $F \in \mathcal{F}$.

$T(i, \neg F \cdot \emptyset, \emptyset, 1)$ is provable from axiom B if and only if F is a tautology for any i in \mathcal{U} and F in \mathcal{F} , by corollaries 1 and 3.

Corollary 4 $\vdash B \rightarrow T(i, \neg F \cdot \emptyset, \emptyset, 1) \equiv \models \rightarrow F$ any $i \in \mathcal{U}$ $F \in \mathcal{F}$.

If $SAT \in \mathcal{P}$ then $T(i, \neg F \cdot \emptyset, \emptyset, 1)$ is provable from axiom B , in polynomial time, for some deterministic Turing machine i in \mathcal{U} any tautology F , by definitions 9 and 12, and corollary 4.

Corollary 5 $SAT \in \mathcal{P} \supset \vdash B \rightarrow T(i, \neg F \cdot \emptyset, \emptyset, 1)$ in $p(F)$ some $i \in \mathcal{U}$ any $F \in TAUT$.

4 Computing time and proof complexity

There is a relationship between computing time⁹ i.e., the number of moves of the tapehead of a Turing machine in a computation, and proof complexity,¹⁰ i.e. the size of a proof in propositional logic. The principal lemma 3 gives this relationship in lemma 4, i.e. if a deterministic Turing machine in \mathcal{U} decides whether or not any sufficiently large negated tautology is satisfiable in polynomial time then there is a proof of the tautology in polynomial size in resolution. An example begins an explanation of this relationship.

Example 1 *If a Turing machine in \mathcal{U} decides whether or not any sufficiently large negated tautology F is satisfiable then the sequent $B \rightarrow T(i, \neg F \cdot \emptyset, \emptyset, 1)$ is provable for any i in \mathcal{U} F in $TAUT$, in Kleene's G_4 . Such a proof can be constructed, first, by successive applications of the rule $\forall \rightarrow$ in G_4 getting the propositional conjunctions (31) to (23) from axiom B . Second, using successive applications of the rule $\supset \rightarrow$ in G_4 . A Turing machine in \mathcal{U} is deterministic, thus the conjunctions (23) - (31) are unique.*

Writing $A(i, F, n)$ for

$$U(\emptyset.1, s_n, z_n, 0, i, i, \emptyset.1) \wedge \quad (23)$$

$$\left(U(\emptyset.1, s_n, z_n, 0, i, i, \emptyset.1) \supset U(x_{n-1}, s_{n-1}, z_{n-1}, q_{n-1}, j_{n-1}, i, \emptyset.1) \right) \wedge \cdots \wedge \quad (24)$$

$$\left(U(x_{m+3}, s_{m+3}, z_{m+3}, q_{m+3}, i, i, \emptyset.1) \supset U(x_{m+2}, s_{m+2}, z_{m+2}, q_{m+2}, j_{m+2}, i, \emptyset.1) \right) \wedge \quad (25)$$

$$\left(U(x_{m+2}, s_{m+2}, z_{m+2}, q_{m+2}, j_{m+2}, i, \emptyset.1) \supset U(x_{m+2}, s_{m+2}, z_{m+2}, q_{m+2}, i, i, \emptyset.1) \right) \wedge \cdots \wedge \quad (26)$$

$$\left(U(x_2, s_2, z_2, q_2, i, i, \emptyset.1) \supset U(x_1, s_1, z_1, q_1, j_1, i, \emptyset.1) \right) \wedge \quad (27)$$

$$\left(U(x_1, s_1, z_1, q_1, j_1, i, \emptyset.1) \supset U(x_1, s_1, z_1, q_1, i, i, \emptyset.1) \right) \wedge \cdots \wedge \quad (28)$$

$$\left(U(x_1, s_1, z_1, q_1, i, i, \emptyset.1) \supset U(\emptyset, \emptyset, \neg F. \emptyset. \emptyset, 1, j_0, i, \emptyset.1) \right) \wedge \quad (29)$$

$$\left(U(\emptyset, \emptyset, \neg F. \emptyset. \emptyset, 1, j_0, i, \emptyset.1) \supset U(\emptyset, \emptyset, \neg F. \emptyset. \emptyset, 1, i, i, \emptyset.1) \right) \wedge \cdots \wedge \quad (30)$$

$$\left(U(\emptyset, \emptyset, \neg F. \emptyset. \emptyset, 1, i, i, \emptyset.1) \supset T(i, \neg F. \emptyset, \emptyset.1) \right) \text{ any } i \in \mathcal{U} \ F \in TAUT \quad (31)$$

$$\begin{aligned} n \in N \text{ some } m \in N \quad & j_m \quad j_{m+2} \quad j_{n-1} \in \mathcal{D} \quad x_m \quad x_{m+2} \quad x_{m+3} \quad x_{n-1} \in L' \\ z_m \quad z_{m+2} \quad z_{m+3} \quad z_{n-1} \in L \quad & s_m \quad s_{m+2} \quad s_{m+3} \in S \quad q_m \quad q_{m+2} \quad q_{m+3} \in Q. \end{aligned}$$

Then,

$$\vdash B \rightarrow T(i, \neg F. \emptyset, \emptyset.1) \supset A(i, F, n) \text{ any } i \in \mathcal{U} \ F \in TAUT \text{ some } n \in N. \quad (32)$$

Not all of $A(i, F, n)$ in (32) is necessary, however, to count the number of moves of the tapehead of a Turing machine in the computation. Informally, writing $V(i, F, n)$ for (23) - (26), and (31). This relation is a simplification of $A(i, F, n)$, but still counts the number of moves of the tapehead of a deterministic Turing machine in the computation.

In example 1, the relation $V(i, F, n)$ counts the number of moves of the tapehead in a computation. It is introduced more precisely as follows.

Definition 13 $V(i, F, n)$ for

$$\left(U(x_0, s_0, z_0, q_0, i, i, \emptyset.1) \supset T(i, \neg F. \emptyset, \emptyset.1) \right) \wedge \quad (33)$$

$$U(\emptyset.1, s_n, z_n, 0, i, i, \emptyset.1) \wedge \quad (34)$$

$$\bigwedge_{1 \leq m \leq n} \left(U(x_m, s_m, z_m, q_m, i, i, \emptyset.1) \supset U(x_{m-1}, s_{m-1}, z_{m-1}, q_{m-1}, j_{m-1}, i, \emptyset.1) \right) \wedge \quad (35)$$

$$\begin{aligned} & \left(U(x_{m-1}, s_{m-1}, z_{m-1}, q_{m-1}, j_{m-1}, i, \emptyset.1) \supset U(x_{m-1}, s_{m-1}, z_{m-1}, q_{m-1}, i, i, \emptyset.1) \right) \\ & \text{any } i \in \mathcal{U} \ F \in TAUT \ n \in N \text{ some } m \in N \quad x_m \quad x_{m-1} \in L' \quad z_n \quad z_m \quad z_{m-1} \in L \\ & \quad s_n \quad s_m \quad s_{m-1} \in S \quad q_m \quad q_{m-1} \in Q \quad j_{m-1} \in \mathcal{D}. \end{aligned}$$

If a Turing machine in \mathcal{U} decides whether or not $\neg F$ is satisfiable in computing time n then $V(i, F, n)$ any $F \in TAUT$ $n \in N$.

Lemma 2 $H(\vdash B \rightarrow T(i, \neg F. \emptyset, \emptyset.1), n) \supset V(i, F, n)$ any $i \in \mathcal{U}$ $n \in N$ $F \in TAUT$.

Proof.³⁸

$$\star \quad H(\vdash B \rightarrow T(i, \neg F \cdot \emptyset, \emptyset \cdot 1), n) \quad i \in \mathcal{U} \quad F \in TAUT \quad n \in N \quad (36)$$

$$\star \quad \vdash B \rightarrow T(i, \neg F \cdot \emptyset, \emptyset \cdot 1), \text{ definition 3} \quad (37)$$

$$\star \quad V(i, F, n), \text{ axiom 1, corollary 2, and definition 13} \quad (38)$$

$$H(\vdash B \rightarrow T(i, \neg F \cdot \emptyset, \emptyset \cdot 1), n) \supset V(i, F, n) \text{ any } i \in \mathcal{U} \quad F \in TAUT \quad n \in N \quad (39)$$

Atoms ("propositional variables") are usually introduced as names for large propositional formulas that are not only inconvenient, but also complex. For sufficiently large input, these atoms give simpler propositional formulas in a standard syntax of propositional logic.

Definition 14 Q_m for $U(x_m, s_m, z_m, q_m, i, i, \emptyset \cdot 1)$ any $i \in \mathcal{U} \quad x_m \in L' \quad z_m \in L \quad s_m \in S \quad q_m \in Q \quad m \in N$ some atom $Q_m \in \mathcal{F}$.

Definition 15 R_{m+1} for $U(x_m, s_m, z_m, q_m, j_m, i, \emptyset \cdot 1)$ any $i \in \mathcal{U} \quad j_m \in \mathcal{D} \quad x_m \in L' \quad z_m \in L \quad s_m \in S \quad q_m \in Q \quad m \in N$ some atom $R_{m+1} \in \mathcal{F}$.

Writing $W(i, F, n)$ for an equivalent formula of $V(i, F, n)$ any $i \in \mathcal{U} \quad F \in TAUT \quad n \in N$, by the auxiliary atoms in definitions 14 - 15.

Definition 16 $W(i, F, n)$ for $Q_n \wedge \bigwedge_{1 \leq m \leq n} (R_m \supset Q_{m-1}) \wedge (Q_m \supset R_m) \wedge (Q_0 \supset T(i, \neg F \cdot \emptyset, \emptyset \cdot 1))$ any $i \in \mathcal{U} \quad F \in TAUT \quad n \in N$ some $m \in N$ some atoms $Q_0, \dots, Q_n, R_1, \dots, R_n \in \mathcal{F}$.

If $V(i, F, n)$ then $W(i, F, n)$ for any $i \in \mathcal{U} \quad F \in TAUT \quad n \in N$, by definitions 13 - 16.

Corollary 6 $V(i, F, n) \supset W(i, F, n)$ any $i \in \mathcal{U} \quad F \in TAUT \quad n \in N$.

Writing $Y(i, F, n)$ for a simplified $W(i, F, n)$.

Definition 17 $Y(i, F, n)$ for $(Q_0 \supset F) \wedge Q_n \wedge \bigwedge_{1 \leq m \leq n} (Q_m \supset Q_{m-1})$ any $i \in \mathcal{U} \quad F \in TAUT \quad n \in N$ some $m \in N$ some atoms $Q_0, \dots, Q_n \in \mathcal{F}$.

If $W(i, F, n)$ then $Y(i, F, n)$ for any $i \in \mathcal{U} \quad F \in TAUT \quad n \in N$, by corollary 1 and definitions 16 - 17.

Corollary 7 $W(i, F, n) \supset Y(i, F, n)$ any $i \in \mathcal{U} \quad F \in TAUT \quad n \in N$.

If a Turing machine in \mathcal{U} decides whether or not $\neg F$ is unsatisfiable in polynomial time then $Y(i, F, n)$, where $n \leq p(F)$, any sufficiently large F in $TAUT$ some n in N .^{39 40}

Lemma 3 $\vdash B \rightarrow T(i, \neg F \cdot \emptyset, \emptyset \cdot 1)$ in $p(F) \supset Y(i, F, n) \wedge n \leq p(F)$ any $i \in \mathcal{U}$ any sufficiently large $F \in TAUT$ some $n \in N$.

³⁸Introduction of a star shall mean introduction of an assumption, and elimination of a star shall mean that an assumption is discharged, cf. Quine 1974 § 37 [17].

³⁹A sentence C is true for sufficiently large natural numbers if there exists a $n' \in N$ such that the sentence $C(n)$ is true for all $n \geq n' \quad n \in N$.

⁴⁰ $\lim_{\substack{|F| \rightarrow \infty \\ F \in TAUT}} \frac{|U(x_m, s_m, z_m, q_m, i, i, \emptyset \cdot 1)|}{|F|} = 1$ and $\lim_{\substack{|F| \rightarrow \infty \\ F \in TAUT}} \frac{|U(x_m, s_m, z_m, q_m, j, i, \emptyset \cdot 1)|}{|F|} = 1$ for $x_m \in L' \quad s_m \in S \quad z_m \in L \quad q_m \in Q \quad i \in \mathcal{U} \quad j \in \mathcal{D} \quad m \in N$, cf. definition 13.

Proof.

$$\star \quad \vdash B \rightarrow T(i, \neg F \cdot \emptyset, \emptyset \cdot 1) \text{ in } p(F) \quad i \in \mathcal{U} \text{ sufficiently large } F \in TAUT \quad (40)$$

$$\star \quad H(\vdash B \rightarrow T(i, \neg F \cdot \emptyset, \emptyset \cdot 1), n) \wedge n \leq p(F) \quad (41)$$

$$\star \quad Y(i, F, n) \text{ some } n \in N, \text{ lemma 2 corollaries 6 – 7 and footnote 40} \quad (42)$$

$$\vdash B \rightarrow T(i, \neg F \cdot \emptyset, \emptyset \cdot 1) \text{ in } p(F) \supset Y(i, F, n) \wedge n \leq p(F) \quad (43)$$

any $i \in \mathcal{U}$ any sufficiently large $F \in TAUT$ some $n \in N$

4.1 Proof complexity

Before the notion of the size of a deduction (proof) is taken up, the idea of a formal deduction in propositional logic is introduced.

The inference rule of the Hilbert system H is modus ponens:

$$\frac{A, A \supset B}{B} \quad (44)$$

where A and B are any propositional formulas.

In the Hilbert system H , a formal deduction is a finite list:

$$F_1, \dots, F_n \quad (45)$$

of formulas, where F_k is either an assumption formula A_1, \dots, A_m , an axiom or follows from F_i and F_j by (44) for $1 \leq i, j < k \leq n$. The formal deduction (45) is a deduction in Hilbert system H of its last formula F_n .⁴¹

In the Hilbert system H , the existence of a formal deduction is written,

$$\Gamma \vdash_H F_n \text{ for some formal deduction } F_1, \dots, F_n \text{ by (44)} \quad (46)$$

where Γ is a list of (zero or more) assumption formulas A_1, \dots, A_m . If there is no assumption, $m < 1$, then there is a formal proof of F_n , i.e. $\vdash_H F_n$.

Hilbert systems are consistent and complete.⁴²

Similarly, for Robinson's resolution systems,¹¹ which have a single inference rule. For the resolution system R the inference rule is:

$$\frac{A \vee P \vee B, \quad C \vee \neg P \vee D}{A \vee B \vee C \vee D} \quad (47)$$

where A, B, C and D are any propositional formulas on disjunctive normal form, DNF, (including the blank formula \sqcup), and P is any propositional atom.

A formal deduction in resolution is like a formal deduction in a Hilbert system, except there is no logical axiom. In resolution system R , a formal deduction is a finite list:

$$F_1, \dots, F_n \quad (48)$$

of formulas, where F_k is either an assumption formula A_1, \dots, A_m , or follows from F_i and F_j by (47) for $1 \leq i, j < k \leq n$. The list (48) is a deduction in resolution system R of its last formula F_n .

Hence, the existence of a formal deduction in resolution system R is written,

$$\Gamma \vdash_R F_n \text{ for there is a formal deduction } F_1, \dots, F_n \text{ by (47),} \quad (49)$$

⁴¹Following Kleene 1967 [12].

⁴²cf. Kleene 1967 [12].

where Γ is a list of (zero or more) assumption formulas A_1, \dots, A_m , if $m < 1$ then $\vdash_R F_n$.

There is a proof rule for indirect deductions (proofs).

$$\text{If } \Gamma, F' \vdash_R \sqcup \text{ then } \Gamma \vdash_R F, \quad (50)$$

where F is a formula on disjunctive normal form, and $F' \equiv \neg F$, where F' is on conjunctive normal form (CNF), and \sqcup is the blank formula.

If Γ is the empty list then there is a proof, i.e. if $F' \vdash_R \sqcup$ then $\vdash_R F$.

Resolution systems are consistent and complete.⁴³

The size of a formal deduction F_1, \dots, F_n is the number of symbols in the formal deduction. Clearly, there is a relation,⁴⁴ which computes the size of a formal deduction in a Hilbert system, and a Robinson resolution system.⁴⁵

Definition 18 $G(\Gamma \vdash_t F_n, u)$ for $u = \sum_{1 \leq k \leq n} |F_k|$ any $u \in N$ some formal deduction F_1, \dots, F_n in t with assumption Γ , where $t \in \{H, R\}$.

A polynomial upper bound of the size of a deduction is introduced.

Definition 19 $\Gamma \vdash_t F$ in $p(F)$ for $G(\Gamma \vdash_t F, u) \wedge u \leq p(F)$ any $F \in \mathcal{F}$ some $u \in N$, where $t \in \{H, R\}$.

Then by lemma 3 and the notion of a formal deduction in resolution system R the following result follows.

If a Turing machine in \mathcal{U} decides whether or not $\neg F$ is satisfiable in polynomial time then there is a polynomial deduction of F in resolution system R for any sufficiently large tautology F on disjunctive normal form.

Lemma 4 $\vdash B \rightarrow T(i, \neg F, \emptyset, \emptyset, 1)$ in $p(F) \supset \vdash_R F$ in $p(F)$ any $i \in \mathcal{U}$ any sufficiently large $F \in TAUT$ on disjunctive normal form.

Proof.

$$\star \quad \vdash B \rightarrow T(i, \neg F, \emptyset, \emptyset, 1) \text{ in } p(F) \text{ } i \in \mathcal{U} \text{ a sufficiently large } F \in TAUT \text{ on DNF} \quad (51)$$

$$\star \quad (Q_0 \supset F) \wedge Q_n \wedge \bigwedge_{\substack{1 \leq k \leq n \\ n \leq p(F)}} (Q_k \supset Q_{k-1}) \text{ some atoms} \quad (52)$$

$$Q_0, \dots, Q_n \in \mathcal{F} \text{ some } n \in N, \text{ lemma 3 and definition 17}$$

$$\star \quad Q_n, \neg Q_n \vee Q_{n-1}, Q_{n-1}, \dots, Q_0, \neg Q_0 \vee F, F, \text{ a deduction in resolution system } R \text{ of } F \text{ by (47) from (52)} \quad (53)$$

$$\star \quad \vdash_R F \text{ in } p(F), \text{ definitions 18 – 19, (49) and (53)} \quad (54)$$

$$\vdash B \rightarrow T(i, \neg F, \emptyset, \emptyset, 1) \text{ in } p(F) \supset \vdash_R F \text{ in } p(F) \text{ any } i \in \mathcal{U} \text{ any sufficiently large } F \in TAUT \text{ on DNF} \quad (55)$$

There are sufficiently large tautologies on disjunctive normal form not having a proof in polynomial time in resolution system R , e.g. valid pigeonhole formulas,¹³ cf. Haken 1985 [8], and Razborov 2003 [18].^{14 15}

⁴³ If $\Gamma \vdash_R F$ then $\Gamma \models F$ (consistency). If $\Gamma \models F$ then $\Gamma \vdash_R F$, where formulas in Γ are on CNF, and F on DNF (completeness), cf. Robinson's theorem 1965 [19].

⁴⁴ $G \subseteq K \times N$. K is the set of first order formal deductions, and N the set of the natural numbers.

⁴⁵ $|F|$ for the number of symbols of $F \in \mathcal{F}$, cf. footnote 36 and definition 4.

Corollary 8 $\neg(\vdash_R F$ in $p(F)$ any sufficiently large $F \in TAUT$ on disjunctive normal form).

SAT is not in \mathcal{P} in the simply consistent extension \mathbf{B}' of theory \mathbf{B} ,^{43 39} by a proof by contradiction.

Theorem 1 $SAT \notin \mathcal{P}$ is true in the simply consistent extension \mathbf{B}' of theory \mathbf{B} .

Proof.

★ $SAT \in \mathcal{P}$ in the simply consistent extension \mathbf{B}' of \mathbf{B} , definition 12 (56)

★ $\vdash B \rightarrow T(i, \neg F . \emptyset, \emptyset . 1)$ in $p(F)$ some $i \in \mathcal{U}$ any (57)
sufficiently large $F \in TAUT$ on DNF, corollary 5

★ $\vdash_R F$ in $p(F)$ any sufficiently large $F \in TAUT$ on DNF, lemma 4 (58)

$SAT \notin \mathcal{P}$ in \mathbf{B}' , contradiction (58) and corollary 8, corollary 2 (59)

In the extension \mathbf{B}'' of theory \mathbf{B}' the formula $\mathcal{P} = \mathcal{N}\mathcal{P}$ is defined by the Cook-Levin theorem.² Then, by theorem 1.

Theorem 2 $\mathcal{P} \neq \mathcal{N}\mathcal{P}$ is true in the simply consistent extension \mathbf{B}'' of theory \mathbf{B} .

By lemma 1, definitions 11 - 12, and theorem 1.

Corollary 9 $SAT \notin \mathcal{P}$ is true, and provable in the simply consistent extension \mathbf{B}' of theory \mathbf{B} .

By lemma 1, theorem 2, and corollary 9.

Corollary 10 $\mathcal{P} \neq \mathcal{N}\mathcal{P}$ is true, and provable in the simply consistent extension \mathbf{B}'' of theory \mathbf{B} .

Acknowledgment

Hanna-Nina Ekelund, Niklas Ekelund, Andreas Hamfelt, Kaj Børge Hansen, Sophie Maisnier-Patin, Erik Nerep, Jørgen Fischer Nilsson, Catuscia Palamidessi, Torsten Palm, Alan Robinson, Thomas Sjöland, and Carl-Anton Tärnlund thank you.

References

- [1] Keith L. Clark and Sten-Åke Tärnlund. A first order theory of data and programs. In Bruce Gilchrist, editor, *Information Processing 77*, volume 7, pages 939–944, Amsterdam, The Netherlands, 1977. North-Holland.
- [2] A. Colmerauer, H. Kanoui, R. Pasero, and P. Roussel. Un Système de Communication Homme-machine en Français. Technical report, Groupe Intelligence Artificielle, Université d’Aix-Marseille, Luminy, 1973.

- [3] Stephen Cook. The complexity of theorem-proving procedures. In *Third Annual ACM Symposium on Theory of Computing*, pages 151–158, New York, NY, USA, 1971. ACM Press.
- [4] Stephen A. Cook. The Importance of the P versus NP Question. *Journal of the ACM*, 50(1):27–29, 2003.
- [5] Stephen A. Cook and Robert A. Reckhow. The relative efficiency of propositional proof systems. *Journal of Symbolic Logic*, 44(1):36–50, 1979.
- [6] Martin Davis. *Computability and Unsolvability*. McGraw–Hill, NY, USA, 1958.
- [7] Gerhard Gentzen. Untersuchungen über das Logische Schliessen. *Mathematische Zeitschrift*, 39:176–210 and 405–431, 1935.
- [8] Armin Haken. The intractability of resolution (complexity). *Theoretical Computer Science*, 39:297–308, 1985. Ph D thesis University of Illinois at Urbana-Champaign 1984.
- [9] J. Hartmanis and R. E. Stearns. On the computational complexity of algorithms. *Transactions of the American Mathematical Society*, 117:285–306, 1965.
- [10] David Hilbert and Paul Bernays. *Grundlagen der Mathematik*, volume 1. Springer-Verlag, Berlin, 1934. Volume 2, 1939.
- [11] Richard M. Karp. Reducibility among combinatorial problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, New York, NY, USA, 1972.
- [12] Stephen C. Kleene. *Mathematical Logic*. John Wiley and Sons, New York, USA, 1967. First corrected printing, March, 1968.
- [13] Robert A. Kowalski. Predicate Logic as a Programming Language. In J.L. Rosenfeldt, editor, *Information Processing 74*, pages 569–574. Amsterdam, The Netherlands, 1974.
- [14] L. Levin. Universal search problems (in Russian). *Problemy Peredachi Informatsii*, 9(3):115–116, 1973.
- [15] Marvin Minsky. *Computation Finite and Infinite Machines*. Prentice-Hall, Inc., Englewood Cliffs NJ, USA, 1967.
- [16] Emil L. Post. Recursive Unsolvability of a Problem of Thue. *Journal of Symbolic Logic*, 12(3):1–11, 1947.
- [17] W. V. Quine. *Methods of Logic*, volume 3. Routledge & Keagan Paul, London, UK, 1974.
- [18] Alexander A. Razborov. Resolution lower bounds for the weak functional pigeon hole principle. *Theoretical Computer Science*, 303(1):233–243, 2003.
- [19] John Alan Robinson. A Machine-Oriented Logic Based on the Resolution Principle. *Journal of the ACM*, 12(1):23–41, 1965.

- [20] Michael Sipser. *Introduction to the Theory of Computation*. Thomson Course Technology, 2005. Second Edition International Edition.
- [21] Steve Smale. Mathematical Problems for the Next Century. *Mathematical Intelligencer*, 20:7–15, 1998.
- [22] Sten-Åke Tärnlund. Horn clause computability. *BIT*, 17(2):215–226, 1977. Cf. TRITA-IBADB-1034, The Royal Institute of Technology 1975, Sweden.
- [23] Sten-Åke Tärnlund. Computing. Updated 2008, Nov 2004.
- [24] Sten-Åke Tärnlund. \mathcal{P} is not equal to \mathcal{NP} . *arXiv e-prints*, October 2008.
- [25] Alan M. Turing. On Computable Numbers with an Application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, 2/46:230–265, 1936.
- [26] David H. D. Warren. *Applied logic - its use and implementation as a programming tool*. Ph.D. thesis, Department of Artificial Intelligence, University of Edinburgh, Edinburgh, UK, 1977. Reprinted in Technical Report 290, 1983, AI Center, SRI International, Menlo Park, CA, USA.